Software Engineering Department

Braude College of Engineering

Final Project – Phase A (61998)

Automated System for University Exam Scheduling

Project Code: 26-1-D-12

By:

Alon Dahan

Karyn Arama

Advisor:

Dr. Julia Sheidin

Dr. Avital Shulner

https://github.com/KarynA19/Exam-Optimizer.git

Table of Contents

Abstract

The final examination scheduling process in academic departments is an extremely complex logistical task, demanding simultaneous compliance with a wide array of conflicting constraints derived from faculty availability, religious and academic calendar restrictions, classroom capacity, and the mandatory institutional rules regarding student examination load. Manually managing this process consumes valuable time resources, increases the risk of scheduling errors (such as overlaps or violations of holy day restrictions), and often results in sub-optimal timetables for both lecturers and students.

This project presents the development of a Computerized Examination Scheduling System, designed to automate this process for the Administrative Assistant to the Head of the Software Engineering Department, while ensuring full adherence to the college's policies and the specific demands of the department. The system utilizes optimization algorithms to generate a comprehensive schedule by implementing a multi-layered constraint model.

The model includes specific consideration for the following factors: distinction between regular faculty members and external lecturers (where only faculty constraints are considered); precise calendar restrictions (holidays, fast days, and fixed dates for Moed A and Moed B periods); mandatory student load rules (up to 3 weekly exams per student); handling of cross-departmental exams (held simultaneously); lecturer constraints (no two exams for the same lecturer in the same week).

The final system offers the user an initial automated scheduling along with the capability for a Re-shuffle based on secondary parameters, providing a flexible, accurate, and immediate solution to the scheduling problem.

1. Introduction

1.1 General Background

Academic institutions, and especially technological colleges with large departments such as the Software Engineering Department, routinely face a complex logistical task: scheduling final semester examinations (Moed A and Moed B periods). This process is not a simple matter of filling days and hours, but rather a multi-variable optimization problem that requires meticulous consideration of the needs and constraints of numerous factors: the academic faculty, the recommended course-taking route and the institutional calendar.

Examination timetables must comply with a long list of "Hard Constraints" whose violation is impossible. These constraints include, among others, religious regulations (fast days, holidays), student load rules (up to 3 weekly exams per student), faculty availability (regular permanent faculty members only), and mandatory fixed times for cross-departmental exams

1.2 The Need for an Automated System

Currently, the scheduling process is performed manually by the department's Administrative Assistant. This process is susceptible to multiple failures:

1. Time and Resources: The process consumes significant working time and requires repeated checks to locate clashes or constraint violations.
2. Risk of Errors: The complexity of the requirements, especially the distinction between permanent faculty and external lecturers, or the consideration of multi-faith fast days increases the risk of scheduling errors that could disrupt the academic order.
3. Lack of Optimality: Manual scheduling often leads to "achievable" solutions that are not optimal in terms of distributing the load on students.

There is a vital need for an automated system that will simplify the complex logistical task, ensure full compliance with all constraints, and provide the Administrative Assistant with a fast and reliable tool for generating the timetable.

1.3 Project Goal

The goal of this final project is to design and develop a Computerized Examination Scheduling System for the Software Engineering Department. The system will be based on an optimization model capable of weighing all the Hard Constraints collected from the Department Head's office, including:

- Constraints related to faculty (ranks, type of position).
- Calendar constraints (holidays, fast days, start and end of the semester, duration of Moed A and B).
- Operational constraints (cross-departmental exams).
- Mandatory student load limitations.

The final system will offer an initial automated scheduling solution and allow for further changes and re-shuffles based on secondary parameters, while maintaining full compliance with all rules. The system will streamline the workflow, reduce errors, and free up valuable administrative time.

2. Literature Review

The University Examination Timetabling Problem (UETP) is a classic NP-hard combinatorial optimization challenge that involves assigning examinations to limited time slots and rooms while satisfying a complex set of constraints. These constraints are categorized into strict logistical rules (Hard Constraints) that determine feasibility, and flexible preferences (Soft Constraints) that define schedule quality. Due to the exponential growth of potential combinations, manual scheduling is mathematically impractical for large departments, necessitating advanced algorithmic frameworks to guarantee a valid and optimized solution (Barták et al., 2010).

2.1 Theoretical Framework: Curriculum-Based Scheduling

The academic problem addressed in this project falls under the specific sub-domain of **Curriculum-Based** timetabling. While standard "Post-Enrollment" models attempt to resolve conflicts for every individual student based on specific registration data, Curriculum-Based models assume that students follow a published "recommended curriculum" or pathway.

Although Bonutti et al. (2012) primarily defined the standard for *Course* Timetabling (CB-CTT), their theoretical framework regarding conflict definition is directly applicable to our *Examination* problem. In this model, two events are considered to be in conflict if they belong to the same recommended semester or year, rather than relying on individual student data. This reduction in problem space allows for more robust optimization of the "Hard Constraints" that guarantee the feasibility of the standard academic path.

2.2 Theoretical Framework: Constraint Satisfaction and Classification

Mathematically, the examination timetabling problem is modeled as a **Constraint Satisfaction Problem (CSP).** *A constraint satisfaction problem consists of:*

- a set of variables X (exams).
- a set of domains D (the time slots and rooms) that can be assigned to each variable.
- a set of constraints C that restrict the allowable combinations of assignments (Barták et al., 2020).

Solving a CSP involves finding a state where every variable in *X* is assigned a value from *D* such that all constraints in *C* are satisfied. In the context of the UETP, the complexity stems from the fact that the set C is not monolithic. Instead, the literature emphasizes a rigorous separation of constraints into a hierarchy: satisfying **Hard Constraints** determines *feasibility*, while optimizing **Soft Constraints** determines *quality* (Burke & Petrovic, 2002).

To solve these NP-hard problems, researchers often utilize techniques like **Integer Linear Programming (ILP)** or **Constraint Programming (CP)**. These methods model binary "yes/no" decisions—such as whether a specific exam is assigned to a specific time slot—to strictly enforce these logical rules (Aslan & Kapanoglu, 2017).

2.2.1 Hard Constraints (Feasibility)

These are the inviolable rules that the system must satisfy to produce a valid schedule. The literature supports several critical hard constraints essential for academic logistics:

- **Gap Constraints:** A mandatory gap of more than three days between exams within the same recommended curriculum ensures students have adequate recovery time. This is a variation of the "period-spread" constraint discussed in standard benchmarks.
- **Prerequisite Dependencies:** A foundational course (e.g., *Algorithms 1*) and its successor (*Algorithms 2*) cannot occur on the same day. This prevents "logical

sequencing" conflicts for students retaking courses, a constraint type formally modeled in Integer Linear Programming (ILP) approaches (Güler et al., 2021).

- **Calendar and Resource Restrictions:** The absolute prohibition of exams on holidays and the limitation of "One Exam per Department on Fridays" act as "masking constraints," effectively removing specific slots from the available search space before optimization begins.

2.2.2 Soft Constraints (Optimization Quality)

Once feasibility is achieved, the quality of the schedule is measured by how well it satisfies flexible preferences.

- **Faculty Workload Balancing:** Faculty constraints, such as a minimum gap between invigilation duties (e.g., >3 days), are often treated as soft constraints. *Elen (2022)* notes that while lecturer constraints are critical, treating them as flexible goals rather than rigid rules prevents the system from becoming "over-constrained" and unsolvable.
- **Special Course Handling:** Dedicating specific days to "High-Failure Rate" courses is a heuristic strategy intended to maximize student success rates, prioritizing these courses in the algorithm's objective function.

2.3 Comparative Analysis of Algorithmic Approaches

Solving the UETP generally falls into three algorithmic categories, each with distinct advantages and disadvantages suitable for different institutional needs.

2.3.1 Heuristics and Graph Coloring

Early approaches utilized graph coloring heuristics, where exams are vertices and conflicts are edges (see Fig.1). The goal is to assign “colors” (time slots) such that no two connected vertices share the same color (Burke & Petrovic, 2002).

- **Advantages:** These methods are computationally inexpensive and can generate a result very quickly.
- **Disadvantages:** They are often “greedy” and prone to getting stuck in local optima, meaning they satisfy Hard Constraints but fail to optimize Soft Constraints effectively (Elen, 2022).

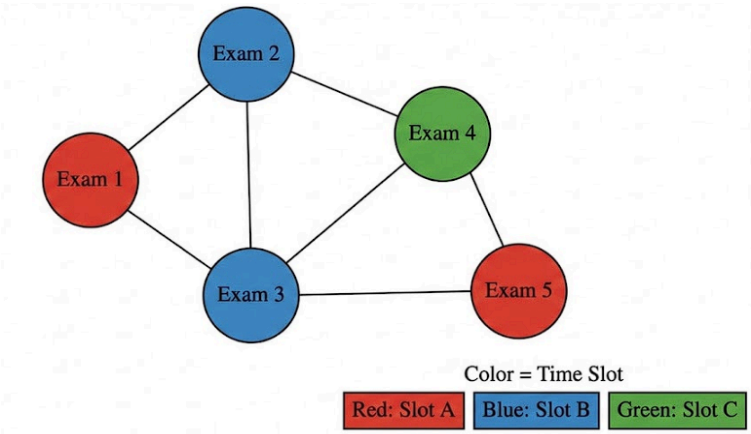


Figure 1: Graph coloring model where vertices represent exams and edges represent conflicts.

2.3.2 Meta-heuristics (Genetic Algorithms)

Meta-heuristics such as Genetic Algorithms (GA), mimic natural evolution by generating a population of schedules and iteratively improving them through crossover and mutation (Fig. 2).

- **Advantages:** They are excellent at navigating vast search spaces and optimizing quality (Soft Constraints) once a feasible solution is found (Pillay & Banzhaf, 2010).
- **Disadvantages:** They do not guarantee finding a feasible solution (one that breaks zero rules) in a fixed time and they require complex parameter tuning (mutation rates, population size) (Elen, 2022).

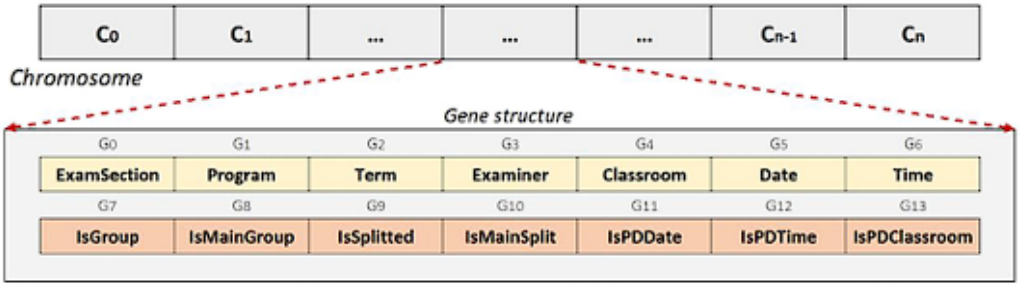


Figure 2: Chromosome encoding used in Genetic Algorithms to represent an exam schedule (Adapted from Elen, 2022).

2.3.3 Exact Methods and Constraint Programming (CP)

Exact methods, such as Constraint Programming (CP), model the problem as a set of logical variables and constraints (Fig. 3).

- **Advantages:** Unlike meta-heuristics, CP is designed to prove feasibility. Lt can guarantee that *if* a valid schedule exists, it will be found, making it ideal for problems with strict Hard Constraints like prerequisite spacing (Güler et al., 2021).
- **Disadvantages:** These methods can suffer from scalability issues on extremely large datasets (NP-hard) though modern solvers like Google OR-Tools have significantly mitigated this risk (Barták et al., 2020).

Recursive BT(*free-variables, instantiation, domains*)

if *free-variables* $= \emptyset$ **then** return *instantiation*
$x \leftarrow$*select-one-variable(free-variables)*
$D_x \leftarrow$ *get-domain*$(x, domains)$
while $D_x \neq \emptyset$ **do**
  $a \leftarrow$ *select-one-value*(D_x)
  if $consistent(instantiation \cup (x, a))$ **then**
    $solution \leftarrow$ **BT**$(free\text{-}variables \backslash \{x\}, instantiation \cup (x, a), domains)$
    if $solution \neq \emptyset$ **then** return *solution*
return $\emptyset$

Figure 3: Backtracking Search (Adapted from Barták et al., 2004).

2.4 Conclusion: Justification for Selected Approach

Based on the comparative analysis of algorithmic approaches, we decided to utilize an **Exact Method (Constraint Programming)** rather than Heuristics or Genetic Algorithms.

While the literature suggests that Hybrid approaches are necessary for massive datasets to manage scalability, the specific requirements of the Software Engineering Department (approx. 50 courses) fall well within the operational capabilities of modern CP solvers like Google OR-Tools.

- **Handling Hard Constraints:** As noted in Section 2.4.3, CP provides a mathematical guarantee of feasibility, which is critical for the “Hard Constraints” defined by the department (e.g., no double-booking faculty
- **Optimization capabilities:** Unlike greedy heuristics which get stuck in local optima, a CP solver can search the entire solution space to optimize "Soft Constraints" (student load balancing) without needing a separate repair stage.

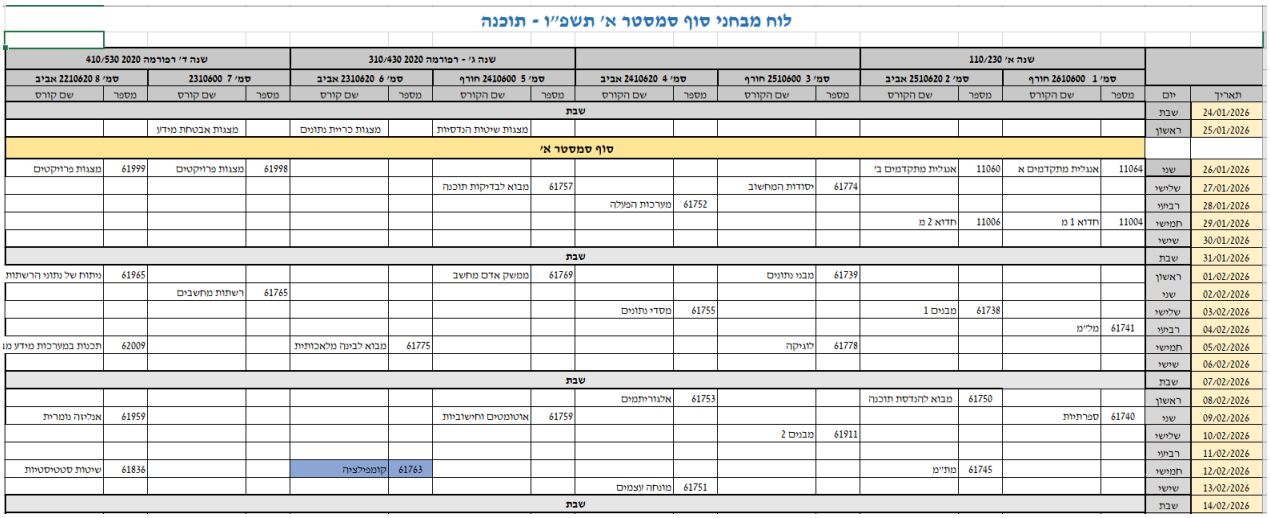
Therefore, this project adopts a **Pure Constraint Programming architecture**, leveraging the "Search and Bound" capabilities of the CP-SAT solver to ensure both schedule validity and quality.

3. Direct User Interaction (Interviews and Observation)

The most crucial step in the process was direct engagement with the key stakeholder, Ora, the Head of Department Assistant responsible for scheduling.

We conducted detailed interviews with Ora to understand the existing problems, pain points, and overall goals for the automated system.

To capture the existing manual heuristics and workflow, we scheduled sessions to observe the entire scheduling process, from start to finish. Ora demonstrated her current methodology, which uses an Excel-based system (Fig. 4) and involves several manual steps:

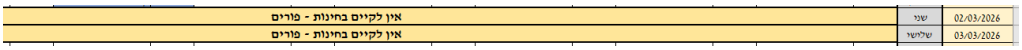


לוח מבחני סוף סמסטר א' תשפ"ו - תוכנה

	שנה א' 110/230						שנה ג' - רפורמה 310/430 2020						שנה ד' רפורמה 410/530 2020					
	סמ' 1 2610600 חורף		סמ' 2 2510620 אביב		סמ' 3 2510600 חורף		סמ' 4 2410620 אביב		סמ' 5 2410600 חורף		סמ' 6 2310620 אביב		סמ' 7 2310600		סמ' 8 2210620 אביב	
תאריך	יום	מספר	שם הקורס	מספר	שם הקורס	מספר	שם הקורס	מספר	שם הקורס	מספר	שם הקורס	מספר	שם הקורס	מספר	שם הקורס	מספר	שם הקורס	מספר
24/01/2026	שבת																	
25/01/2026	ראשון								מצגות שיטות הנדסיות		מצגות כריית נתונים		מצגות אבטחת מידע					
26/01/2026	שני	11064	אנגלית מתקדמים א	11060	אנגלית מתקדמים ב'													
27/01/2026	שלישי					61774	יסודות המחשוב			61757	מבוא לבדיקות תוכנה			61998	מצגות פרויקטים	61999	מצגות פרויקטים	
28/01/2026	רביעי							61752	מערכות הפעלה									
29/01/2026	חמישי	11004	יחידה 1 מ	11006	יחידה 2 מ													
30/01/2026	שישי																	
31/01/2026	שבת																	
01/02/2026	ראשון					61739	מבני נתונים			61769	ממשק אדם מחשב					61965	ניתוח של נתוני הרשתות	
02/02/2026	שני													61765	רשתות מחשבים			
03/02/2026	שלישי			61738	מבנים 1			61755	מסדי נתונים									
04/02/2026	רביעי	61741	מלי"מ															
05/02/2026	חמישי					61778	לוגיקה					61775	מבוא לביונה מלאכותית			62009	תכנות במערכות מידע מו	
06/02/2026	שישי																	
07/02/2026	שבת																	
08/02/2026	ראשון			61750	מבוא להנדסת תוכנה			61753	אלגוריתמים									
09/02/2026	שני	61740	ספרתיות							61759	אוטומטים וחישוביות					61959	אנליזה נומרית	
10/02/2026	שלישי					61911	מבנים 2											
11/02/2026	רביעי																	
12/02/2026	חמישי	61745	פוז"מ									61763	קומפילציה			61836	שיטות סטטיסטיות	
13/02/2026	שישי							61751	מוטיבה ועצמים									
14/02/2026	שבת																	

Figure 4: Example of Schedule

- **Calendar Exclusion:** Marking off holidays and fast days from the schedule.



אין לקיים בחינות - פורים	שני	02/03/2026
אין לקיים בחינות - פורים	שלישי	03/03/2026

- **Sequencing and Prioritization:** Initial scheduling based on internal knowledge, such as ensuring wide spacing for exams with high failure rates (an implicit soft constraint).
- **Faculty Constraints:** Manually ensuring a gap for each lecturer.
- **Post-Processing:** Entering the final, approved schedule into the Gilboa system one by one.

During the observation, follow-up questions were asked to clarify specific decision-making processes, particularly regarding the trade-offs involved in scheduling conflicts and the priority given to Staff members' constraints over others.

- Does the scheduling strictly follow the recommended academic track?
- What is your preferred format for viewing the generated schedule?

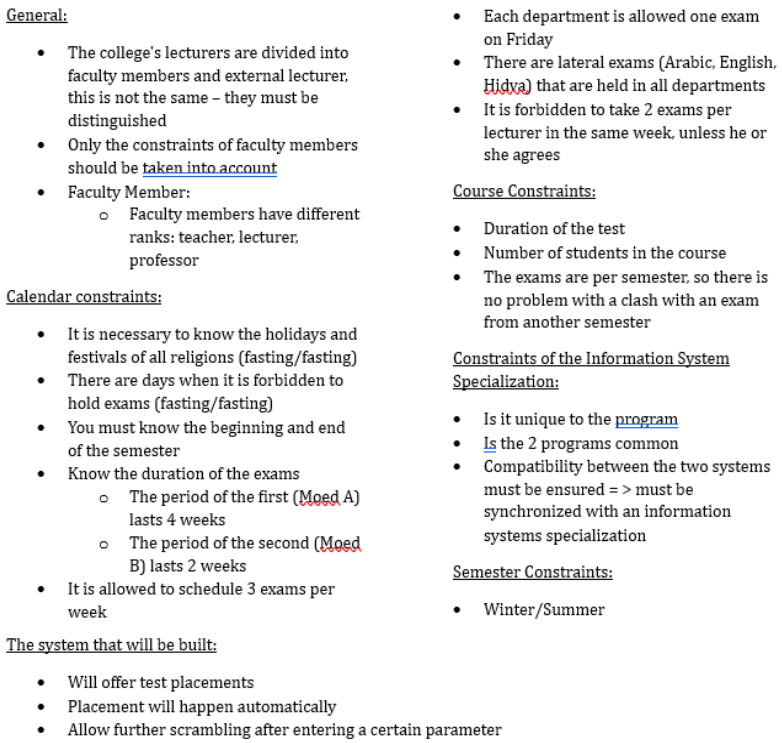
- How do you currently manage the scheduling process? (Request for a live demonstration)
- Which constraints are prioritized during the initial phase (e.g., minimum 2-semester gap)?
- How is information regarding faculty availability and preferred dates reported to you?
- When and how are exam dates for cross-departmental courses (e.g., English, Hebrew) received and integrated?

3.1 Document Analysis

The formal framework of the project was derived from analyzing the provided requirements document (the "Ora Requirements" document, Fig 5), which defined all the essential **hard constraints** and entities.

Defining Constraints: The document analysis established the rigid scheduling rules:

- Duration of exam periods (4 weeks for Moed A and 2 weeks for Moed B).
- Maximum number of exams per week (3).
- Faculty categorization (Staff vs. Adjunct) and the specific requirement to adhere strictly to Staff member constraints.
- Logistical rules (e.g., maximum one Friday exam per department).
- Handling of cross-departmental exams (e.g., Arabic, English, Calculus) which must be held simultaneously.



General:

- The college's lecturers are divided into faculty members and external lecturer, this is not the same – they must be distinguished
- Only the constraints of faculty members should be taken into account
- Faculty Member:
  - Faculty members have different ranks: teacher, lecturer, professor

Calendar constraints:

- It is necessary to know the holidays and festivals of all religions (fasting/fasting)
- There are days when it is forbidden to hold exams (fasting/fasting)
- You must know the beginning and end of the semester
- Know the duration of the exams
  - The period of the first (Moed A) lasts 4 weeks
  - The period of the second (Moed B) lasts 2 weeks
- It is allowed to schedule 3 exams per week

The system that will be built:

- Will offer test placements
- Placement will happen automatically
- Allow further scrambling after entering a certain parameter
- Each department is allowed one exam on Friday
- There are lateral exams (Arabic, English, Hidya) that are held in all departments
- It is forbidden to take 2 exams per lecturer in the same week, unless he or she agrees

Course Constraints:

- Duration of the test
- Number of students in the course
- The exams are per semester, so there is no problem with a clash with an exam from another semester

Constraints of the Information System Specialization:

- Is it unique to the program
- Is the 2 programs common
- Compatibility between the two systems must be ensured = > must be synchronized with an information systems specialization

Semester Constraints:

- Winter/Summer

Figure 5: Ora's Requirements

Defining Entities: The analysis formalized the required data entities (Fig 6), including Course details (duration, student count, type) and the need for synchronization between the Software Engineering and Information Systems programs.

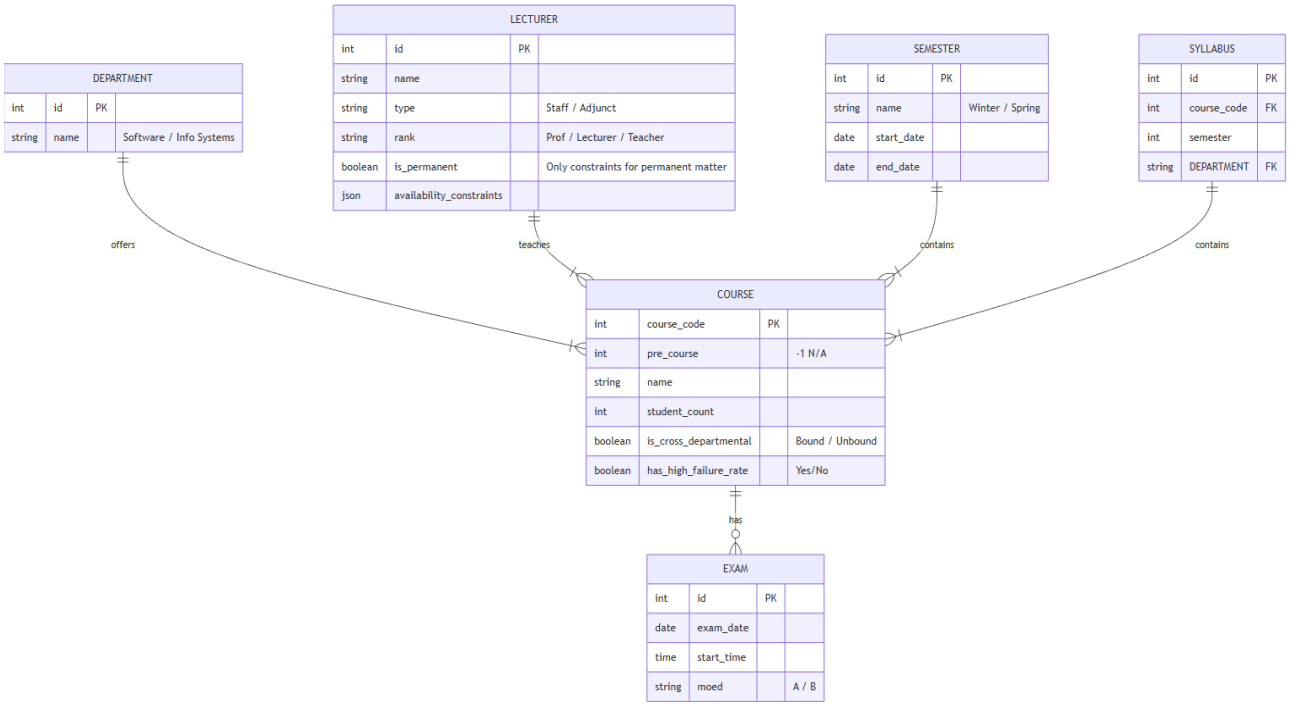


Figure 6: Entity-Relationship Diagram

This combined approach ensures that the system is not only functionally compliant with all written rules but is also practically usable and effective for the end-user's day-to-day workflow.

4. Development Process

The development of the Computerized Examination Scheduling System (CES) follows a structured approach that prioritizes the complex logistical constraints of the department while ensuring a flexible user experience. We employ a phased methodology, transitioning from human-centered requirements gathering to a robust mathematical optimization architecture.

4.1 The Process

The development process comprises all four phases of the UCD approach: (1) understanding the context, (2) specifying user requirements, (3) designing solutions, and (4) evaluating against requirements (Shneiderman & Plaisant, 2010). Phase 4 is explained in more detail in section 7.

4.1.1 Understanding the Context

To foster a deep understanding of the problem domain, we analyzed the current manual scheduling process, which is characterized by disorganized data and heavy reliance on fragmented spreadsheets.

- **Stakeholder Interaction:** The most crucial step was direct engagement with **Ora**, the Head of Department Assistant responsible for scheduling. We conducted detailed interviews to understand pain points, such as the risk of scheduling errors during multi-faith fast days.
- **Structured Observation:** We observed the existing manual methodology, which involves manual calendar exclusion for holidays, sequencing based on internal knowledge (such as spacing high-failure rate courses), and manual entry into the **Gilboa** system.
- **Document Analysis:** The formal framework was derived from the **"Ora Requirements"** document, which established the rigid boundaries of the academic calendar (4-week Moed A, 2-week Moed B) and mandatory student load limits (max 3 exams per week).

4.1.2 Specifying User Requirements

Based on stakeholder interaction and document analysis, we identified the following functional and non-functional requirements.

Functional Requirements (FR)

ID	Requirement	Category
FR1	The system shall allow the Administrator to define and mark specific dates as "blocked" (e.g., holidays, fast days) to prevent exam scheduling on these days.	Configuration & Input
FR2	The system shall provide an interface to classify faculty members by rank and toggle constraint applicability (e.g., enable/disable constraints for external lecturers).	Configuration & Input
FR3	The system shall execute a scheduling algorithm to generate a complete exam timetable for Moed A and Moed B.	Scheduling Algorithm

FR4	During schedule generation, the system shall validate that no student group exceeds the configurable limit of exams per week (default: 3).	Validation Rules
FR5	The system shall verify that no lecturer is assigned to more than one exam per week and alert the user if a conflict is detected without a manual override.	Validation Rules
FR6	The system shall identify courses flagged as "Shared" or "Cross-Departmental" and force simultaneous scheduling for these exams across all relevant department schedules.	Scheduling Algorithm
FR7	Re-shuffle Mechanism: The system shall allow the user to manually lock specific exam slots (fix a date/time for a course) and re-execute the scheduling algorithm for the remaining unscheduled exams.	Flexibility & Optimization

Non-Functional Requirements (NFR)

ID	Requirement	Category
NFR1	**Execution Time:** The automatic algorithm shall be required to produce a Valid Schedule within **up to 60 seconds** from activation.	Performance
NFR2	**Usability:** The User Interface shall be intuitive and tailored for a non-technical user (the Administrative Assistant), allowing for a clear presentation of the schedule and potential conflicts.	User Interface

NFR3	**Security and Privacy:** Access to the system shall be granted only through user authentication, and sensitive data (such as faculty details) shall be stored securely.	Security
NFR4	**Maintainability:** The system shall be designed with modularity to allow for easy updates to scheduling rules and the addition of new constraints in the future.	Code Quality

4.1.3 Designing Solutions - Prototype

The system features three primary interconnected modules developed to simulate the administrative workflow:

- **Constraints & Settings View:** Allows the administrator to set the Moed periods, toggle holiday avoidance, and define student load (Figs 7,8,9).

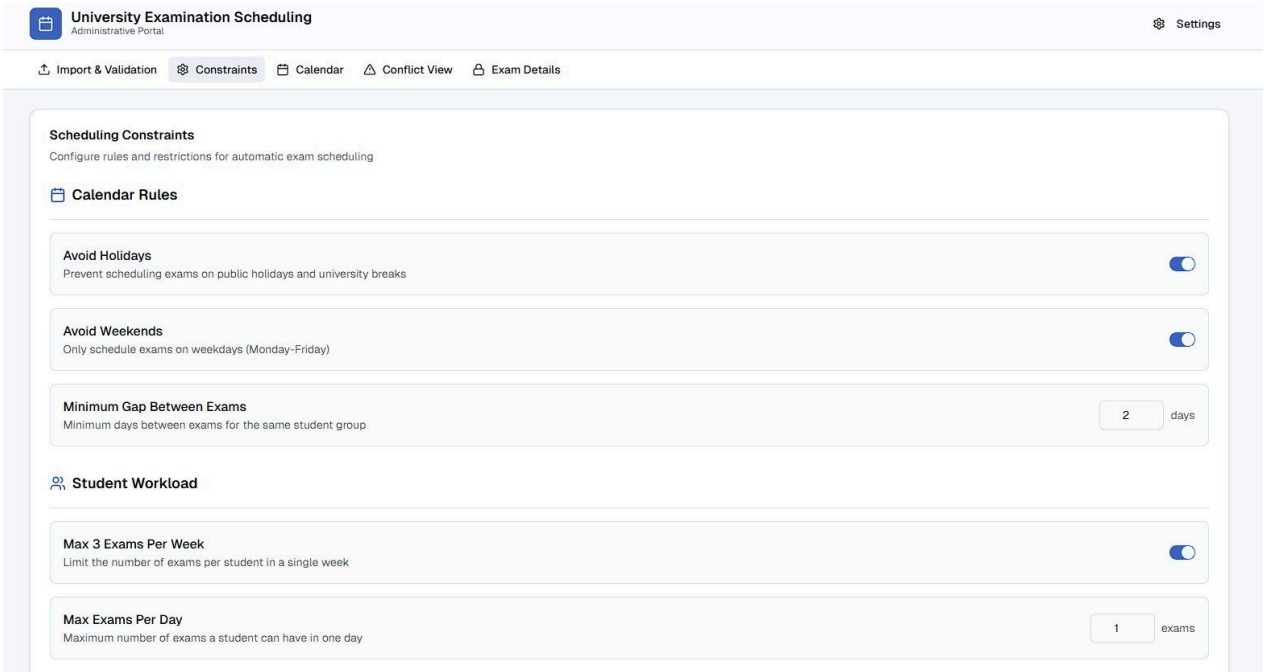


Figure 7: Constraint Alignment

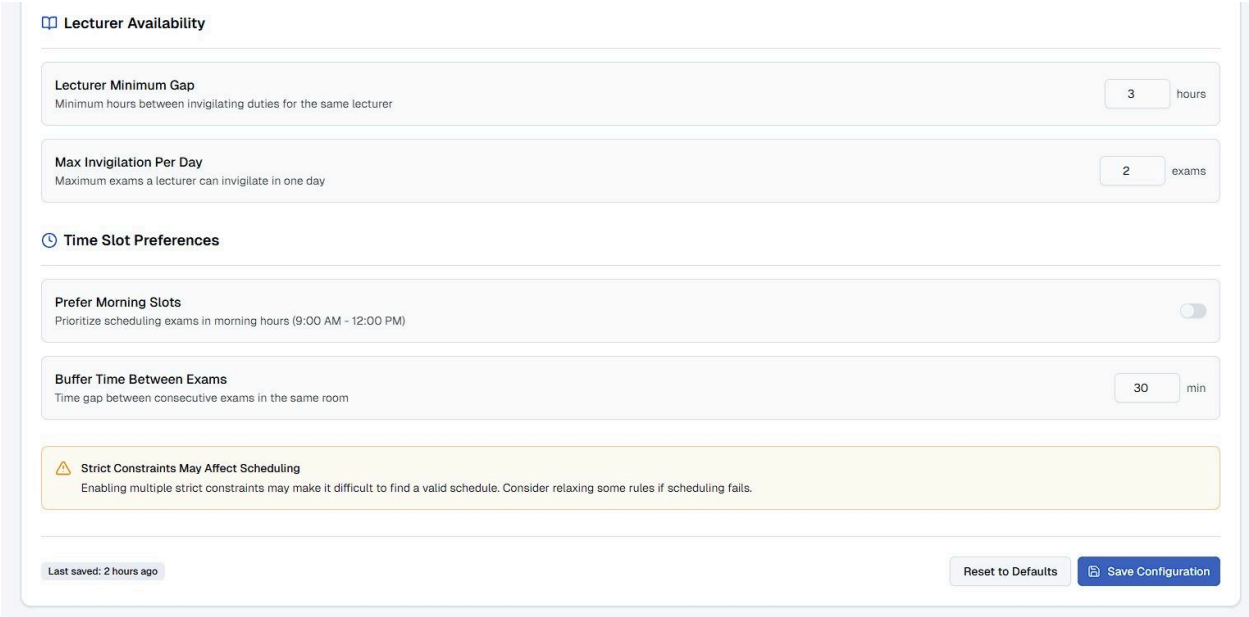


Figure 8: Lecturer Alignment

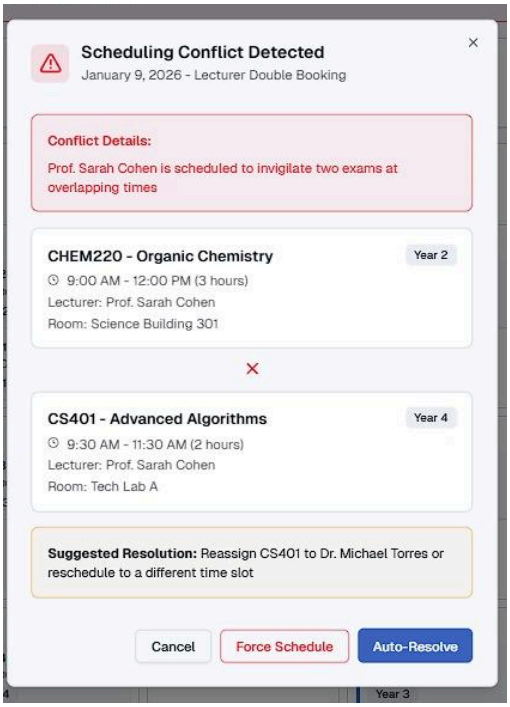


Figure 9: Conflict Preview

- **Data Management Portal:** Supports bulk uploads of teacher constraints and course information via CSV or Excel (Fig 10).

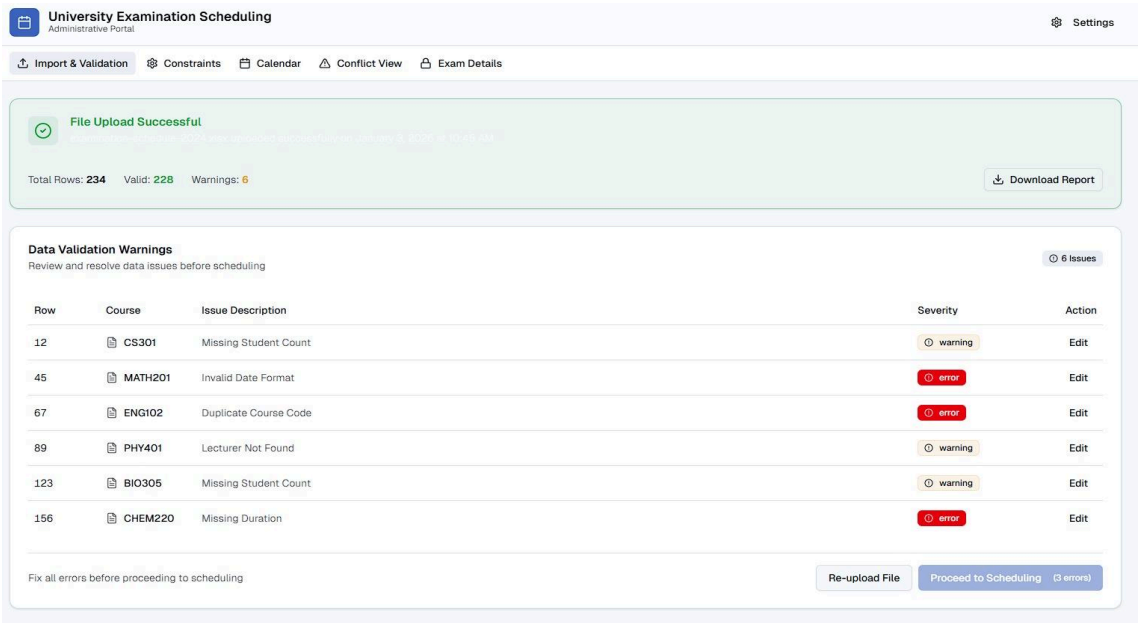


Figure 10: File Upload of constraints and preview of warnings.

- **The Main Scheduler:** A dynamic interface where scheduled and unscheduled exams are presented alongside a monthly calendar, supporting the manual **Re-shuffle** capability. (Fig 11)

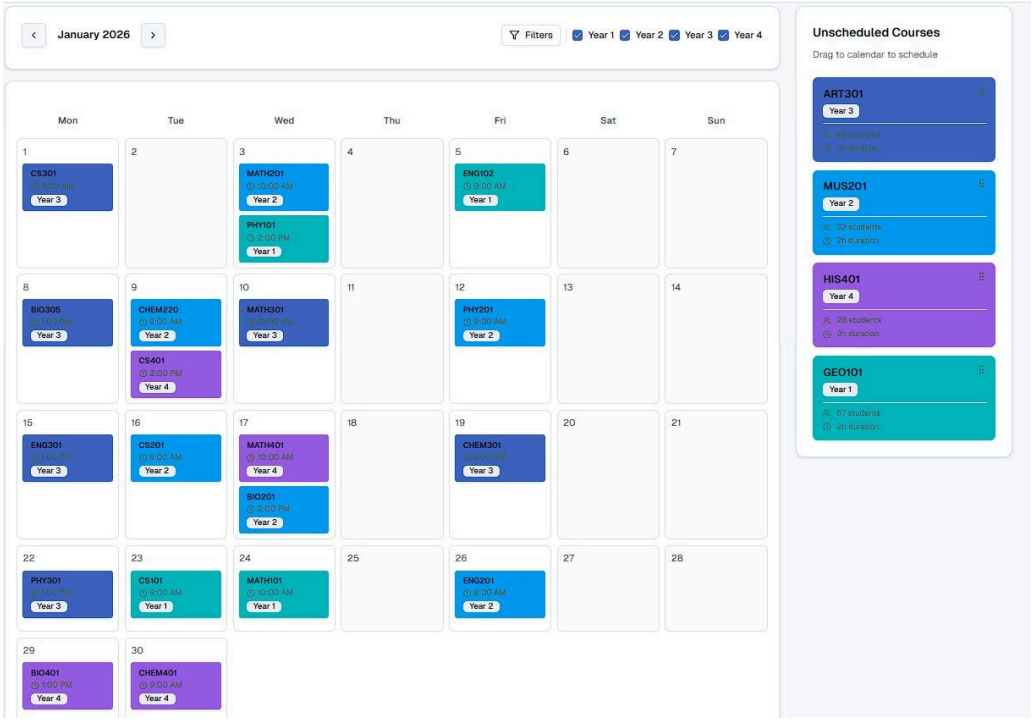


Figure 11: Main Month Schedule preview

4.2 Architecture

4.2.1 Core Technologies and Tools

The choice of technologies was driven by the need for robust mathematical optimization and rapid web deployment. These technologies power the base to our system (Fig 12).

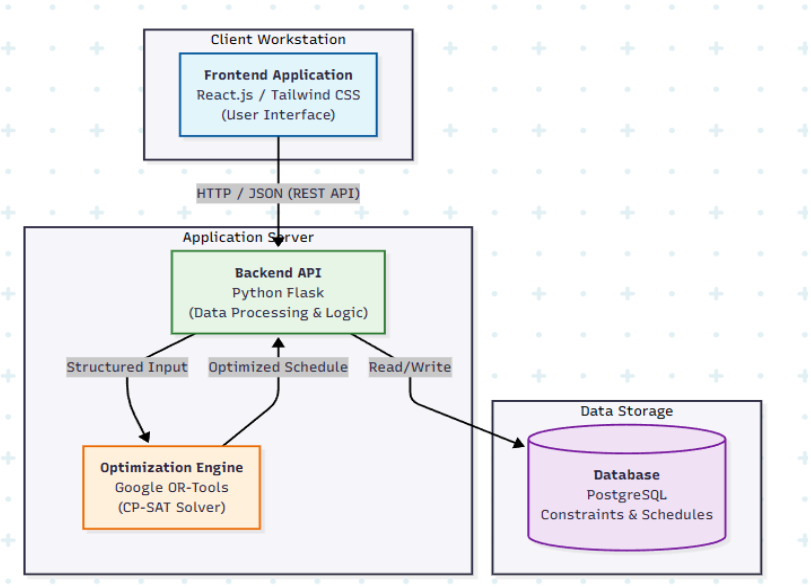


Figure 12: Technology Diagram

Technology	**Role in Project**	**Justification**
Python[1]	Primary Programming Language	Python offers extensive libraries for data manipulation and optimization, making it the industry standard for operations research tasks.
Google OR-Tools[2]	Optimization Solver	An open-source software suite for optimization. Specifically, its **CP-SAT Solver** (Constraint Programming) is ideal for scheduling problems with complex, discrete constraints like ours (e.g., "no two exams for the same lecturer").

Pandas[3]	Data Processing	Used for efficient parsing, cleaning, and structuring of the input data (Excel files) before feeding it into the optimization model.

[1] Python Software Foundation. (n.d.). *Welcome to Python.org*. https://www.python.org/ [2] Google. (n.d.). *OR-Tools | Google Developers*. https://developers.google.com/optimization [3] The pandas development team. (n.d.). *pandas documentation*. https://pandas.pydata.org/

Backend Development (Server Side)

The backend serves as the bridge between the user interface and the algorithmic core.

Technology	Role in Project	Justification
Flask (Python)	Web Framework	A lightweight micro-framework that allows for quick setup of API endpoints. It integrates natively with the Python-based optimization core, eliminating the need for complex inter-process communication.
REST API	Communication Protocol	Standard architectural style for communication between the Client and Server, ensuring modularity and scalability.

Frontend Development (User Interface)

The frontend provides the interactive environment for the Administrative Assistant.

Technology	Role in Project	Justification
React.js	Frontend Framework	A JavaScript library for building user interfaces. Its component-based architecture is perfect for managing

		the dynamic state of the schedule view (e.g., dragging and dropping exams).
Tailwind CSS	Styling Framework	A utility-first CSS framework that allows for rapid UI development with a clean, modern aesthetic, ensuring the system is visually accessible and professional.
Axios	HTTP Client	Used to send asynchronous HTTP requests to REST endpoints and perform CRUD operations (Create, Read, Update, Delete) on the scheduled data.

Data Management

Reliable storage is essential for maintaining course lists, faculty constraints, and generated schedules.

Technology	**Role in Project**	**Justification**
PostgreSQL	Database	An advanced, open-source object-relational database system. Its reliability and support for complex queries make it suitable for storing the relational data of the university's entities (Students, Courses, Faculty).
JSON / CSV	Data Interchange Formats	CSV is used for the initial bulk data import (from the department's Excel files), while JSON is used for lightweight data exchange between the Frontend and Backend.

5. System Operational Flow

The end-to-end workflow emphasizes **User-Centered Design** (Fig 13), where the algorithm provides an optimal starting point but final authority rests with the user **(Shneiderman & Plaisant ,2010)**.

1. **Data Ingestion:** Uploading constraints and raw department data.
2. **Validation:** Checking data validity and generating error reports if inconsistencies are found.
3. **Two-Stage Optimization:**
    - **Stage 1: Constructive Heuristic** to satisfy all hard constraints.
    - **Stage 2: CP Optimizer** to refine the schedule based on soft preferences.
4. **User Review & Re-shuffle:** The administrator reviews the schedule. If unsatisfied, a manual re-shuffle allows for locking specific slots and recalculating the remaining assignments.
5. **Final Export:** Once approved, the data is exported in a format compatible with the **Gilboa** system.

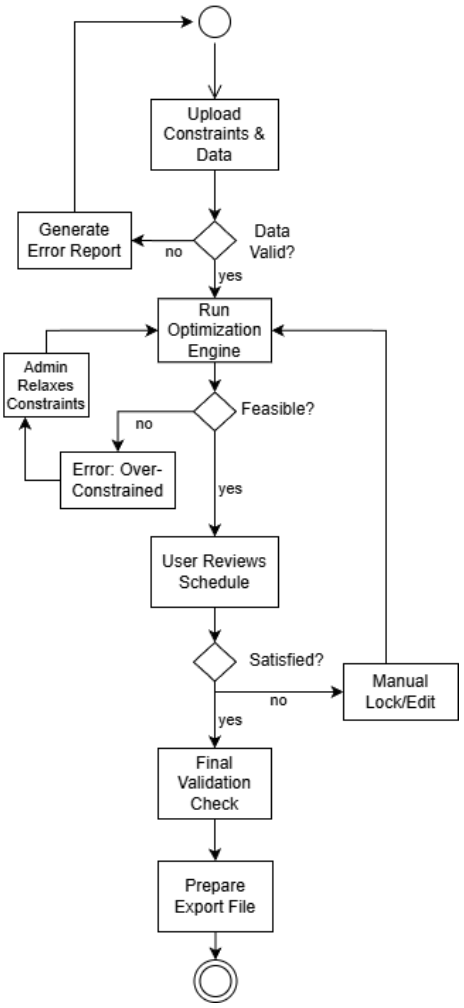


Figure 13: Activity Diagram displaying User Centered Design.

6. Expected Challenges

Based on our initial research and requirements gathering, it is clear that developing the Computerized Examination Scheduling System involves significant technical and logistical hurdles. Transitioning from a manual, experience-based process to a fully automated algorithmic solution presents several key challenges we need to address.

- **Algorithmic Complexity and Performance:** The core problem we are solving is NP-hard, meaning the number of possible schedule combinations is massive due to "combinatorial explosion." A major technical challenge will be implementing an algorithm efficient enough to navigate this huge search space and generate a valid schedule within our 60 second performance requirement.
- **Modeling Implicit Human Knowledge:** During our interviews with the administrator, we learned that many scheduling decisions depend on "implicit heuristics" human intuition that isn't officially written down, such as knowing which difficult courses need extra spacing between exams. Translating this qualitative knowledge into precise mathematical constraints for the engine will be difficult.
- **External Dependencies and Synchronization:** The requirement to synchronize "cross-cutting exams" (like Calculus or English) poses a significant risk. Because these are controlled by external departments, a sudden schedule change on their end could instantly invalidate our entire departmental timetable.
- **Implementing the "Re-shuffle" Capability:** The feature allowing users to manually lock specific slots and re-run the schedule is computationally complex. The challenge is ensuring the algorithm can quickly detect if a user's manual lock creates an impossible scenario for the remaining exams and communicate that clearly to the user.

7. Evaluation Plans

7.1 Testing Plan

This plan verifies that the Computerized Examination Scheduling System (CES) meets technical specifications and provides real-world utility for the department administrator, addressing the requirements for validation and success criteria.

The testing process is divided into four main phases:

Phase 1: Data Import and Integrity Tests

Goal: Verify correct parsing and filtering of input data based on departmental logic. This phase confirms that the system accurately parses the uploaded Excel file, correctly classifies faculty (prioritizing permanent staff constraints over external lecturers), filters out elective courses as required, and identifies shared "bound" courses needing synchronization vs. unique "unbound" courses.

Phase 2: Hard Constraint Verification (Feasibility)

Goal: Validate strict adherence to the non-negotiable "Iron Rules." Testing ensures the system never violates critical constraints: enforcing a mandatory 4-day gap between a lecturer's exams, honoring specific lecturer blackout dates and all religious holidays, ensuring simultaneous slots for cross-departmental exams, and limiting Fridays to a maximum of one exam per department.

Phase 3: Soft Constraints and Optimization Quality

Goal: Verify the system generates a "smart" schedule optimized for student success. This phase assesses the quality of the schedule by verifying that the algorithm maximizes study gaps leading up to identified "high-failure rate" courses, prioritizes buffers between prerequisite and advanced courses, and efficiently schedules non-conflicting unlinked courses in parallel.

Phase 4: Performance and Usability Testing

Goal: Ensure the system meets speed requirements and is usable for the administrator. Testing confirms that the system can generate a valid schedule for a standard 50 course dataset within ~60 seconds. It also validates the "Re-Shuffle" capability, ensuring the engine can correctly regenerate the schedule around manually locked slots without breaking hard constraints.

7.2 User Evaluation

The user will evaluate the system by performing a full scheduling workflow using realistic departmental data. Following hands-on usage, the user will complete a **System Usability Scale (SUS)** questionnaire along with several additional qualitative questions to assess usability, clarity, and overall satisfaction with the system (Brooke, 1995).

7.3 Key Metrics for Evaluation

The system's success will be measured against the following quantitative algorithmic goals and qualitative user experience targets for the Department Administrator.

Quantitative Metrics (Algorithmic Performance)

- **Feasibility Rate:** Achieve **100% compliance** with all "Hard Constraints. Any single violation (e.g., double-booking faculty or scheduling on a Fast Day) renders the schedule invalid, detected by automated validation scripts.
- **Execution Efficiency:** Generate a complete, valid initial schedule for a standard semester (approx. 50 courses) in **≤ 60 seconds**.
- **Soft Constraint Optimization:** Demonstrate improved schedule quality (e.g., better study gaps between exams) compared to historical manual baselines.

- **Re-Shuffle Responsiveness:** Regenerate a valid schedule following a manual user adjustment (e.g., locking a slot) in **≤ 30 seconds**.

Qualitative Metrics (User Experience & Workflow)

- **Usability Score (SUS):** Achieve a System Usability Scale score **> 80** based on a standard survey completed by the Administrator (Ora) after testing.

8. GenAI Usage Discloser

During the preparation of this work, the authors used Gemini and ChatGPT to enhance language and grammar. After using this service, the authors reviewed and edited the content as needed and took full responsibility for the content of the published article.

9. References

Aslan, A., & Kapanoglu, M. (2017). A binary integer programming model for exam scheduling problem with several departments. *Beykent University Journal of Science and Engineering*, 10(2), 24-34. https://www.researchgate.net/publication/318589460_A_Binary_Integer_Programming_Model_For_Exam_Scheduling_Problem_With_Several_Departments

Barták, R., Salido, M. A., & Rossi, F. (2010). New trends in constraint satisfaction, planning, and scheduling: a survey. The Knowledge Engineering Review, 25(3), 249–279. https://www.researchgate.net/publication/220254280_New_trends_in_constraint_satisfaction_planning_and_scheduling_A_survey

Bonutti, A., De Cesco, F., Di Gaspero, L., & Schaerf, A. (2012). Benchmarking curriculum-based course timetabling: A survey and proposed framework. *Annals of Operations Research*, 194(1), 59–74. https://www.researchgate.net/publication/220461352_Benchmarking_curriculum-based_course_timetabling_Formulations_data_formats_instances_validation_visualization_and_results

Brooke, J. (1995). SUS - A quick and dirty usability scale.

Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R. (2010). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176(1), 177–192. https://www.researchgate.net/publication/222549395_A_graph-based_hyper-heuristic_for_educational_timetabling_problems

Burke, E. K., & Petrovic, S. (2002). Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2), 266–280.

https://www.researchgate.net/publication/222567018_Recent_research_directions_in_automated_timetabling

Elen, A. (2022). A Complete Solution to Exam Scheduling Problem: A Case Study. *Journal of Scientific Reports-A*, 49, 12–34. https://www.researchgate.net/publication/361716715_A_Complete_Solution_to_Exam_Scheduling_Problem_A_Case_Study

Güler, M. G., Geçici, E., Köroğlu, T., & Becit, E. (2021). A web-based decision support system for examination timetabling. *Expert Systems with Applications*, 183, 115363. https://www.researchgate.net/publication/352290506_A_Web-Based_Decision_Support_System_for_Examination_Timetabling

Pillay, N., & Banzhaf, W. (2010). An informed genetic algorithm for the examination timetabling problem. *Applied Soft Computing*, 10(2), 457–467. https://www.sciencedirect.com/science/article/abs/pii/S1568494609001331

Shneiderman, B., & Plaisant, C. (2010). Designing the user interface: Strategies for effective human-computer interaction. Pearson Education India. https://www.researchgate.net/publication/240953629_Designing_the_user_interface_strategies_for_effective_human-computerinteraction